Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Analysis Report**



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2.

Curso 2023 – 2024

|  |  |
| --- | --- |
| **Fecha** | **Versión** |
| 5/03/2024 | v1.0.0 |

|  |  |  |
| --- | --- | --- |
| **Grupo: C1.011** | | |
| **Autores por orden alfabético** | **Rol** | **Descripción del rol** |
| Abouri, Mohamed - Y7156458E | Developer | Persona encargada de desarrollar el código. |
| Cabello Ranea, Rafael -32094496C | Operador | Encargado de las tareas de campo, de las instalaciones y del mantenimiento de los sistemas de la empresa. |
| Calderón Rodríguez, Francisco Javier - 78233219F | Project Manager | Persona encargada de tomar decisiones de diseño y vigilar el correcto desarrollo |
| Delgado Pallares, David - 29519510E | Tester | Persona encargada de realizar pruebas sobre el código. |
| Ramírez Lara, Iván- 77852382G | Developer | Persona encargada de desarrollar el código. |

**Control de Versiones**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 05/03/2024 | v1.0.0 | Desarrollo de la primera versión. |
| 06/03/2024 | v1.1.0 | Mejoras tras la sesión de feedback con el profesor. |
|  |  |  |
|  |  |  |

**Índice de contenido**

[1.](#_heading=h.gjdgxs) Resumen ejecutivo 4

2[.](#_heading=h.gjdgxs) Introducción 4

3. Análisis de los requisitos 4

4[.](#_heading=h.30j0zll) Evaluación 4

5[. Conclusiones](#_heading=h.1fob9te) 6

**1. Resumen ejecutivo**

Este documento tiene el objetivo de hacer una retrospectiva de todas las tareas que hemos establecido que vamos a realizar en el planning report. Esta retrospectiva nos servirá para descubrir qué tareas han ido bien, mal o regularmente. De esta forma podremos llevar a cabo un plan de mejora con el objetivo de evitar estos errores en entregas posteriores.

**2. Introducción**

Habiendo realizado ya todas las tareas que hemos establecido del sprint planning, es hora de ver cómo ha ido el desarrollo de estas y qué plan de mejora podemos establecer para el siguiente deliverable, evaluando los pros y los contras.

**4. Análisis de los requisitos**

1. **Requisito:** Code audits are essential pieces to ensure the quality of a project. The system must store the following data about them: a code (pattern “[A-Z]{1,3}-[0-9]{3}”, not blank, unique), an execution date (in the past), a type (“Static”, “Dynamic”), a list of proposed corrective actions (not blank, shorter than 101 characters), a mark (computed as the mode of the marks in the corresponding auditing records; ties must be broken arbitrarily if necessary), and an optional link with further information.

* Decisión:
  + La lista de corrective actions se ha implementado mediante un String.
  + A la hora de calcular el atributo mark, se ha estimado que la moda no era una buena representación de las notas de un audit, debido a que un audit con notas AAAFF tendría una A como calificación global, poseyendo dos F significativas, por lo tanto, se ha decidido almacenar como un String, el cual nos de las notas de todos los auditing records asociados.
* Pros:
  + No hace falta resolver empates.
  + Expresa de una mejor manera las calificaciones del audit sin incurrir en calificaciones que no reflejan totalmente los auditing records asociados.
* Contras:
  + Es menos intuitivo a la hora de mostrarse para el usuario del programa.

1. **Requisito:** The result of each code audit is based on the analysis of their audit records. The system must store the following data about them: a code (pattern “AU-[0-9]{4}-[0-9]{3}”, not blank, unique), the period during which the subject was audited (in the past, at least one hour long), a mark (“A+”, “A”, “B”, “C”, “F”, or “F-”), and an optional link with further information.

* Decisión:
  + Para representar el periodo, hemos decidido dividir el atributo en 2 atributos: periodStart, que representa la fecha de inicio del periodo, y periodEnd, que representa la fecha de fin del periodo.
  + El atributo mark se ha representado mediante un enumerado.
* Pros:
  + Se puede determinar la duración del periodo haciendo un cálculo simple.
* Contras:
  + No se almacena directamente la duración, sino que hay que calcularla.

1. **Requisito adicional:** There is a new project-specific role called auditor, which has the following profile data: firm (not blank, shorter than 76 characters), professional ID (not blank, shorter than 26 characters), a list of certifications (not blank, shorter than 101 characters), and an optional link with further information.

* Decisión:
  + La lista de certificaciones se ha implementado como un String.

**4. Evaluación**

* En esta entrega, hemos realizado primero los requisitos opcionales tales como el diagrama UML o las partes de la documentación que han sido posibles.
* Pros:
  + Tener claro cómo son las relaciones entre las entidades para implementarlo de manera correcta.
  + Apoyarnos en el feedback del profesor en el esquema UML antes de desarrollar los requisitos de código.
* Contras:
  + Al esperar al feedback del profesor, disponemos de muy poco tiempo para la implementación de código y en caso de tener que solucionar errores, tenemos muy poco margen.
* A diferencia del entregable 1, hemos corregido el problema de dejar algunas tareas para el final del entregable como requisitos opcionales de documentación que nos sirven para planificar mejor la realización de tareas a lo largo del Sprint.
* En segundo lugar, en este entregable no se han llevado a cabo correctamente las revisiones de las tareas antes de aceptar una pull request directa hacia la rama principal (rama master). Por tanto, se ha pasado código incorrecto o que no funciona a la rama principal y se ha tenido que emplear un tiempo extra para solucionar esos errores.
* Para que esto no ocurra en los siguientes sprints, hemos desarrollado un plan de mejora en el que hemos establecido que se tiene que estar seguro de que la clase está perfectamente implementada y que no tiene ningún error que se tenga que corregir antes de hacer el merge con la rama master. Se crea una rama adicional de desarrollo a partir de la rama master de la que saldrán las distintas ramas para cada una de las tareas y en la que se probará todo el código realizado antes de pasar a la rama master, que se pasará una vez finalizada todas las tareas.
* Pros:
* Cuando se acepte la pull request a la rama master se habrá testeado 2 veces, para pasar de la rama feature a la rama de desarrollo, y para pasar de la rama desarrollo a la rama master, estando seguro de que la implementación es correcta.
* Contras:
  + Tenemos que tomar un poco más de tiempo en planificar cómo hacer la implementación para que no de error y tardaremos más tiempo en hacer merge con las nuevas características
* Además, se añadirá un revisor más a la proteción de la rama master y de la rama de desarrollo, para que no se pueda mergear ninguna pull request sin que al menos 2 personas hayan revisado y aceptado los cambios realizados.
* Pros:
  + Existe mucha más seguridad a la hora de pasar código a la rama principal, para que no se produzcan errores que puedan producir pérdidas de tiempo a los desarrolladores.
* Contras:
  + Los miembros del equipo tienen que emplear más tiempo en revisar el código realizado por sus compañeros.

**4.** [**Conclusiones**](#_heading=h.1fob9te)

En este entregable hemos analizado el plan de mejora que implementamos y hemos concluido que ha sido de gran ayuda para realizar las diferentes tareas del entregable. Con respecto a los problemas que hemos encontrado en este entregable, podemos decir que no han sido graves, ya que solo nos ha causado un poco de retraso, al no ser muy complicado arreglar esos errores. Para intentar reducir esto, en el siguiente entregable usaremos el plan de mejora y lo analizaremos para ver cómo ha funcionado.